Status Update @ New York

IMAGES

Integrated Modeling for Analysis and Generation of Embedded Software

Raj Rajkumar & Bruce Krogh

Carnegie Mellon University

Approved for Public Release, Distribution Unlimited

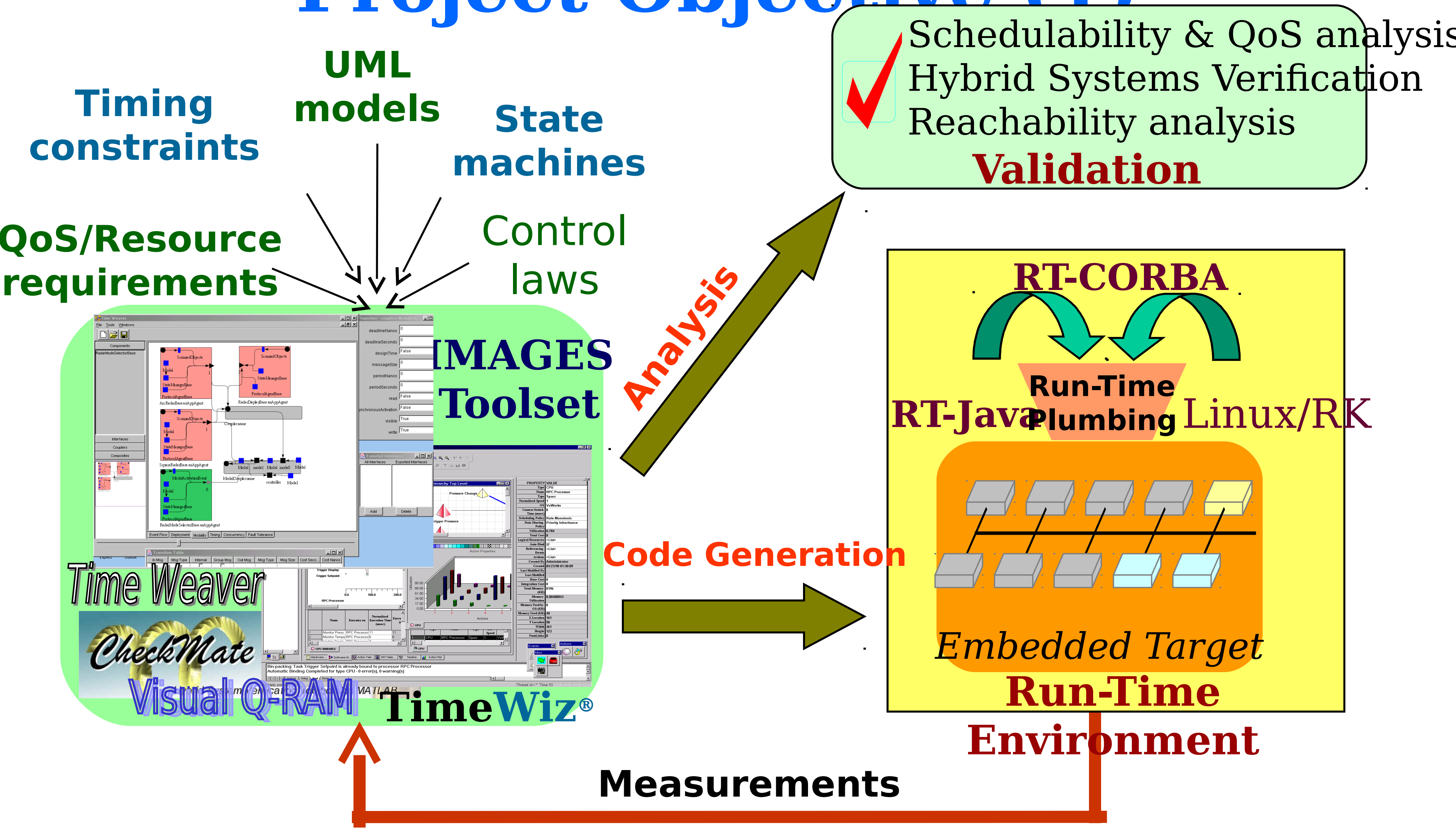
Administrative

Project Title:	**IMAGES: Integrated Modeling for Analysis and Generation of Embedded Software**
PM:	**Dr. John Bay**
PI:	**Profs. Raj Rajkumar and Bruce Krogh**
Co-PIs:	**Prof. Ed Clarke (CMU CS), Dr. Peter Feiler (CMU Software Engineering Institute) and Prof. John Lehoczky (CMU Statistics)**
PI(s) Phone No. and e-mail:	**(412) 268-8707 ; raj@ece.cmu.edu** **(412) 268-2472 ; krogh@ece.cmu.edu**
Company/Institution:	**Carnegie Mellon University**
Contract Number:	**F33615-00-C-1701**
Award End Date:	**June 22, 2003**

Subcontractors and Collaborators

Subcontractors and their roles	• None
MoBIES Non-OEP Collaborators	• Lockheed Martin Aerospace, Upenn, Teknowledge and Vanderbilt
Non-OEP Collaboration Goals	• LM: Applicability of IMAGES tools • Vanderbilt: Interchange format • Penn: Model-checking coordination • Teknowledge: UML Interface tool support
SEC Collaboration Efforts	• None (so far but open)

Problem Description and Project Objective (1)



Timing constraints

UML models

State machines

QoS/Resource requirements

Control laws

MAGES Toolset

Time Weaver

CheckMate

Visual Q-RAM

TimeWiz®

Analysis

Schedulability & QoS analysis
Hybrid Systems Verification
Reachability analysis

Validation

Code Generation

RT-CORBA

Run-Time Plumbing

RT-Java

Linux/RK

Embedded Target

Run-Time Environment

Measurements

Multi-View Modeling + Analyses + Targeting Freedom + *Reusable* Embedded Software

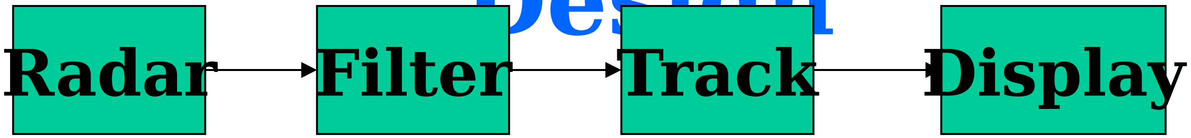
Problem Description and Program Objective (2)

OEP	Avionics	Automotive
Technical capabilities	• End-to-end Timing and **Schedulability analysis** • **Concurrency modeling** ✓**Deployment control** ✓**Event dependency modeling** ✓**Composition of multiple dimensions** • **Fault tolerance modeling**	• End-to-end **Timing analysis** • **RTOS** environment modeling and optimization • Off-line **QoS trade-offs** and optimization
Tool capabilities	• Inter-operability with **UML** and Timing Analysis tools ✓**Multiple views** ✓**XML-based data interchange** ✓**Configurator capabilities**	• Interoperability with Modeling and Analysis Tools • **OSEK**-target code generation
Success Criteria	• Reusability of real-time software • Lower resource utilization	• Timing predictability • OSEK target optimization

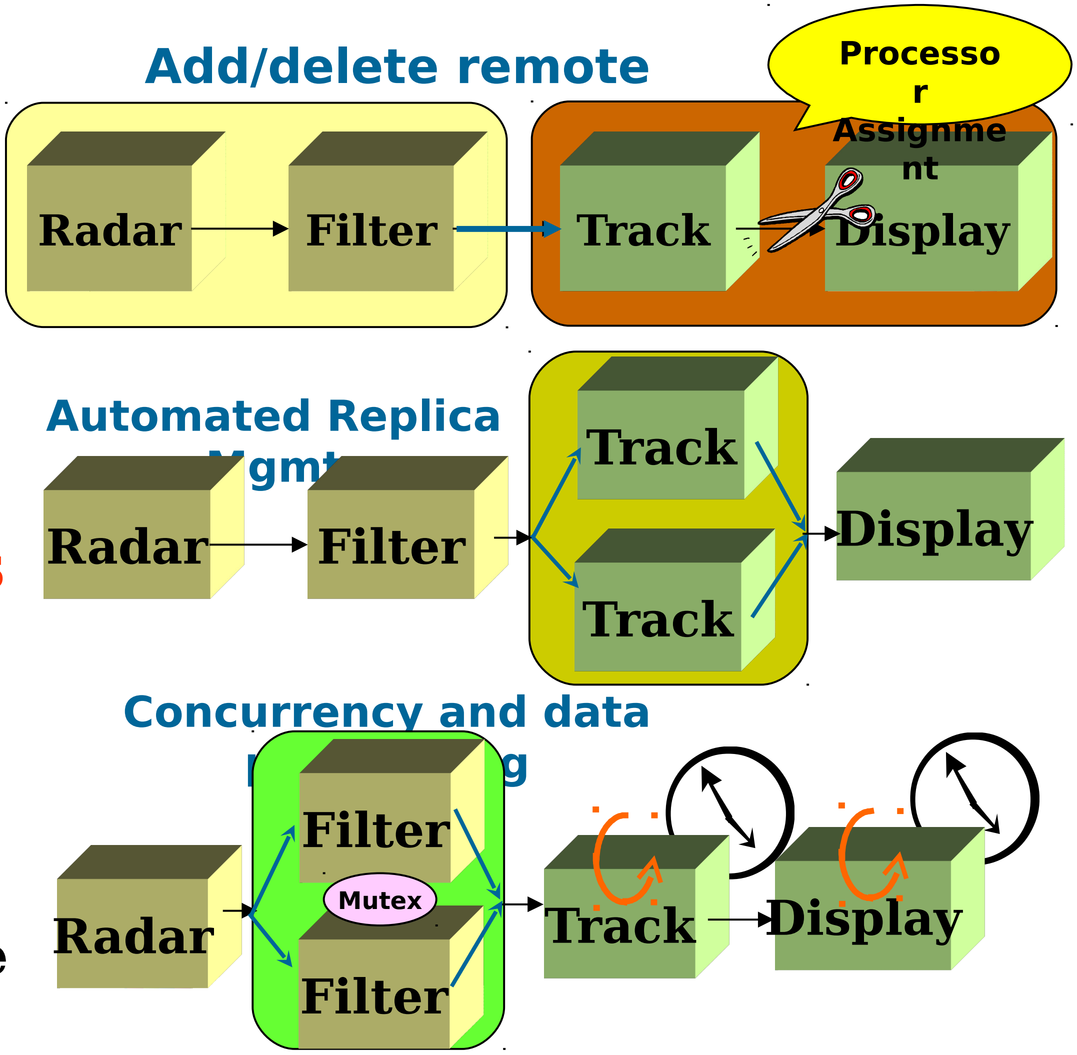
Tool Descriptions

- Time Weaver
  - A framework and tool for creating reusable embedded software components for distributed real-time systems
  - Capture, analyze and optimize non-functional aspects
    - Optimize inter-process communications and degree of concurrency
    - Customize timing parameters
    - Manipulate data path, control path and timing paths independently
  - Generate fully functional target code by linking with functional code segments
- TimeWiz
  - End-to-end timing analysis tool for distributed RT systems
- Visual Q-RAM

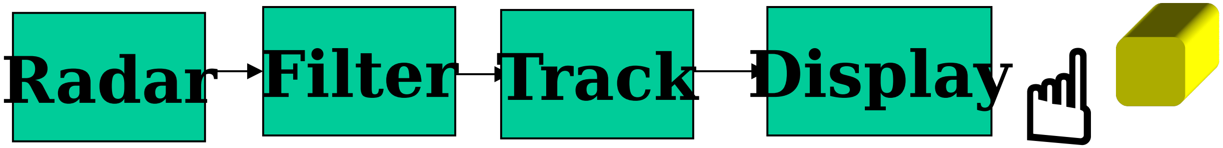
Embedded System Design



- **Functional aspects (application functionality)**
  - Computational activities that transform data from input to output
- **Deployment aspects (system infrastructure)**
  - Connect components
  - Distribute load
  - Specialized hardware
  - Reliability
  - Increase parallelism

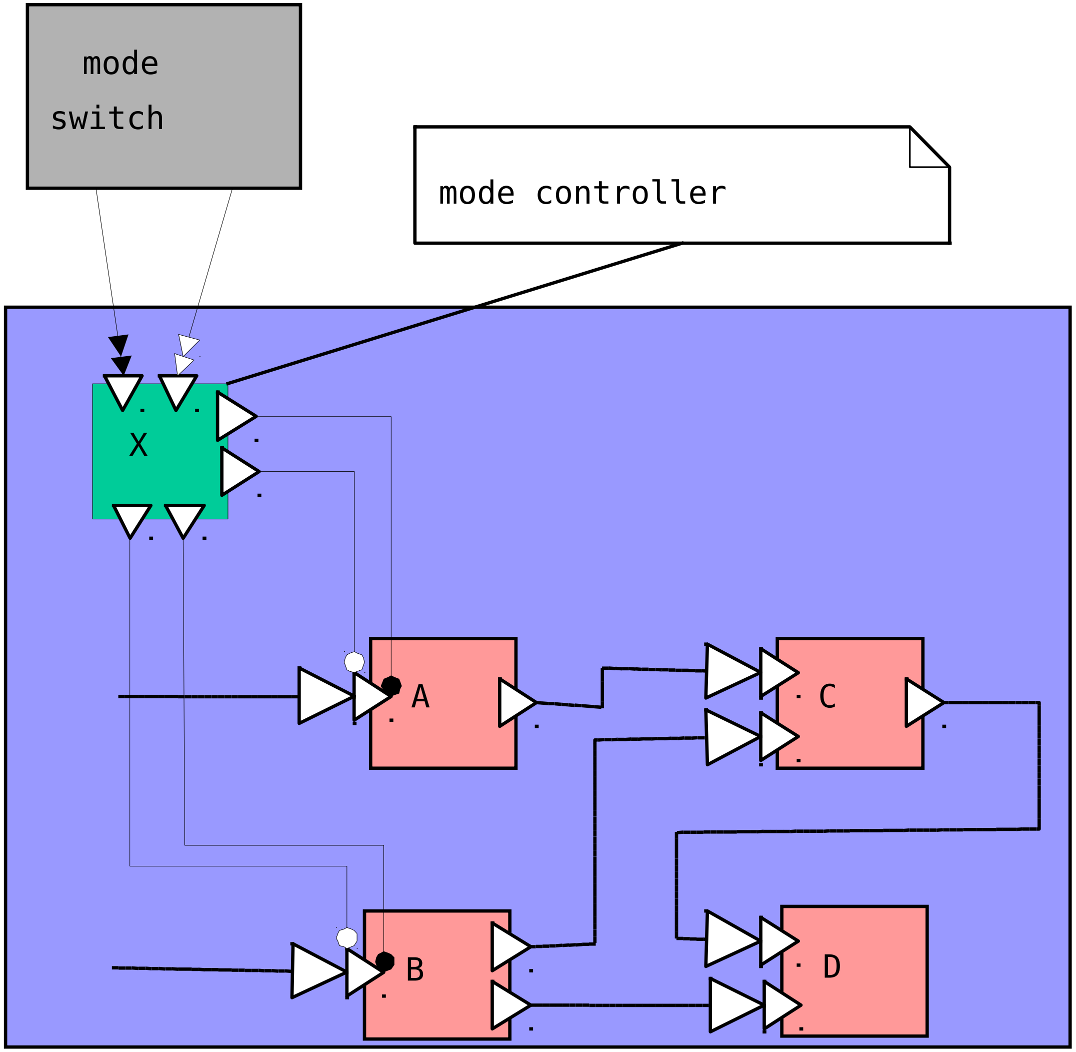


Modality Support



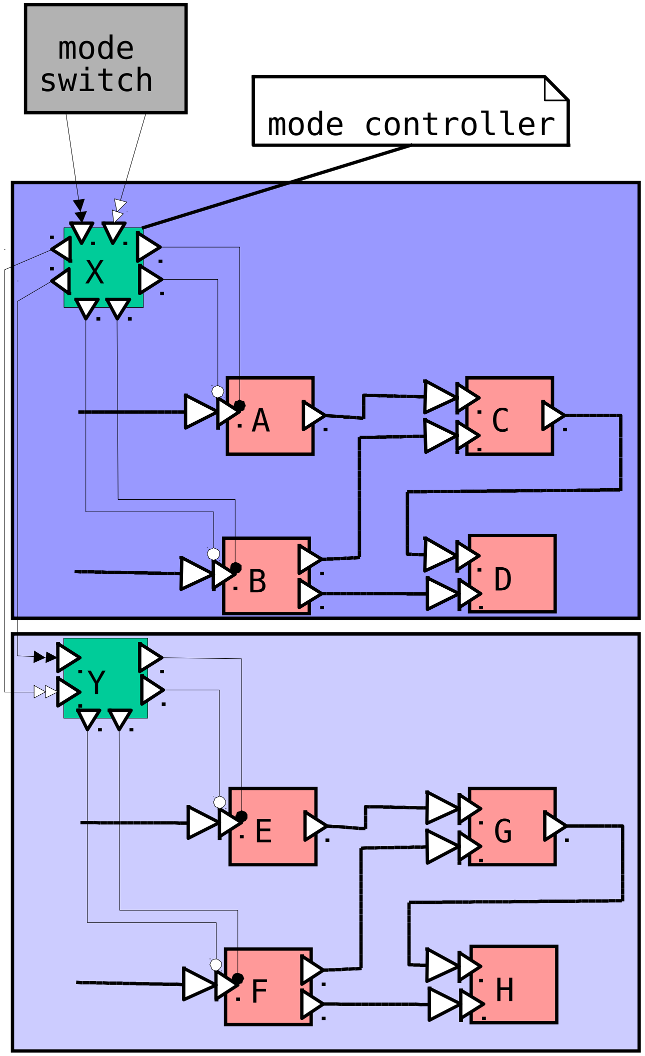
- Challenge:
  - There are potentially hundreds or thousands of such “component modes”
  - The system can move between different system modes
    - E.g. from “attack mode” to “fast escape” mode
- Benefit:
  - Model various modes explicitly
    - Correct code generation
  - Perform worst-case timing analysis across the entire system
    - Current state of the art: use

Modality Primitives



Sub-system 1

Modality Composition

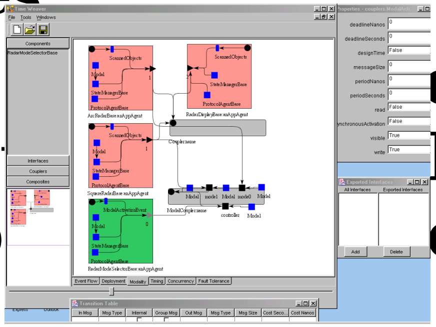


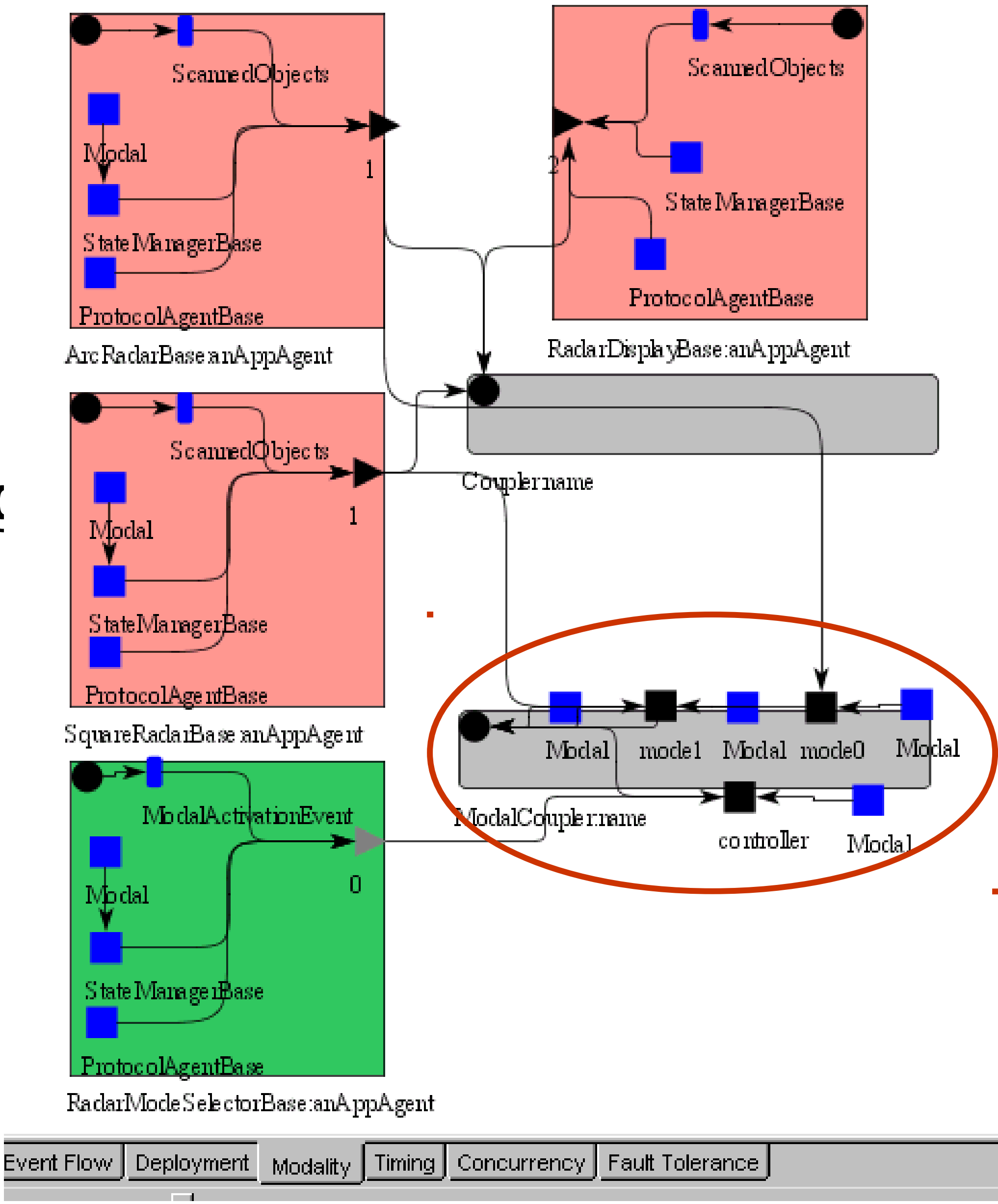
Sub-system 1

Sub-system 2

Modality View

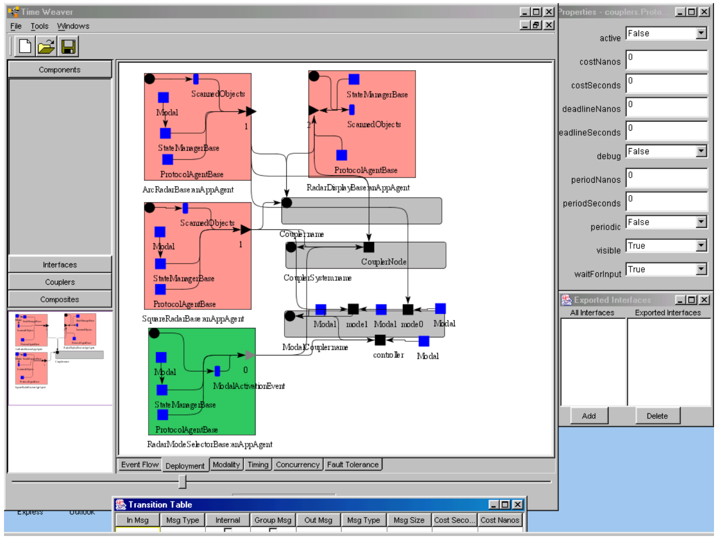
- **Explicitly model and control which components are enabled/disabled.**
  - Conditional execution can also be represented
- **Complete understandin and characterization of the “worst-case” behavior of the system.**
  - Not just “representative” cases
- **Code of targe h comp es.**



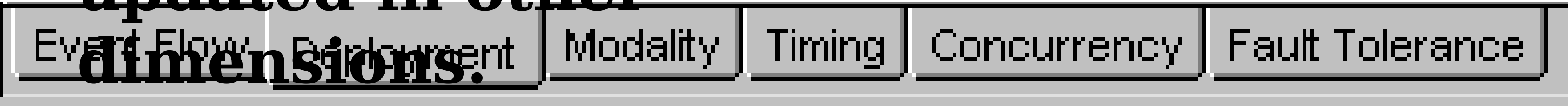


Semantic Orthogonality

- Orthogonal semantics are operated in different views
- Impact of changes (if any) in one dimension are automatically updated in other dimensions:
  - e.g. replication and processor binding

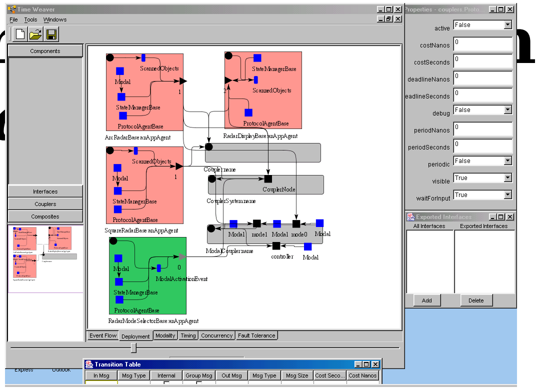


active: False
costNanos: 0
costSeconds: 0
deadlineNanos: 0
deadlineSeconds: 0
debug: False
periodNanos: 0
periodSeconds: 0
periodic: False
visible: True
waitForInput: True



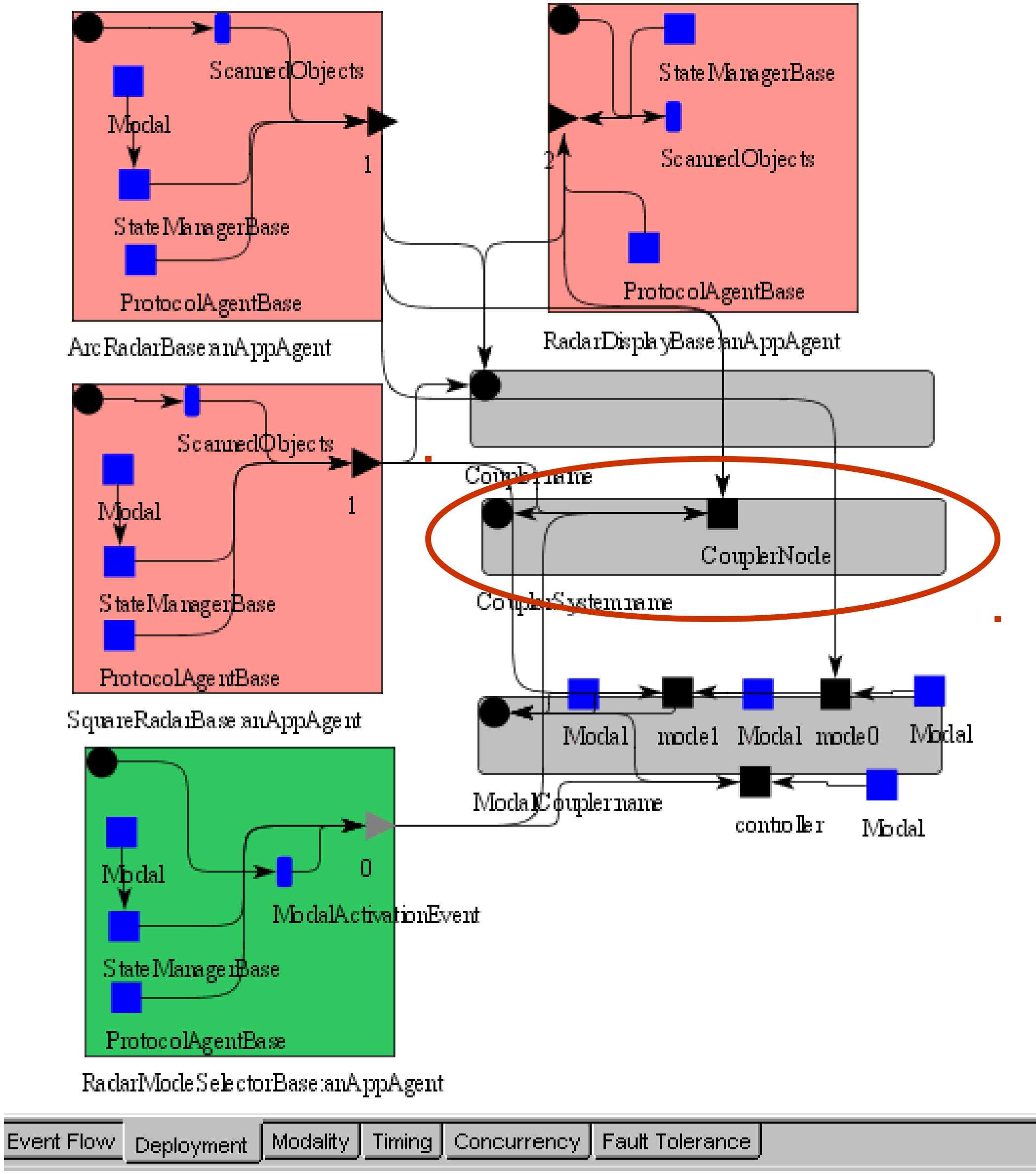
Deployment View

- **Model component bindings to physical processors**
  - **Bindings will be done automatically based on timing analysis (*later*)**
- **Replication of components will place automatic constraints on pr ngs (*l***



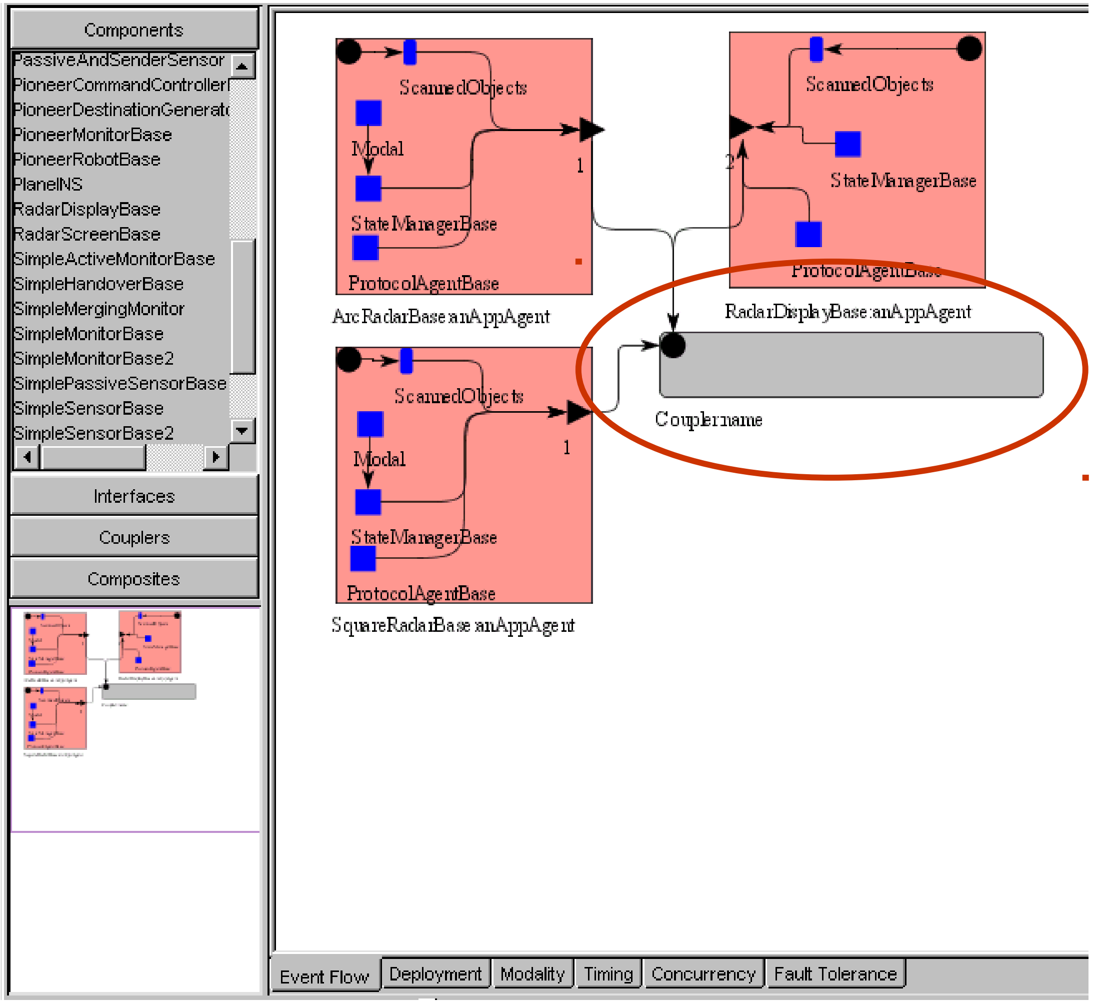
active False
costNanos 0
costSeconds 0
deadlineNanos 0
deadlineSeconds 0
debug False
periodNanos 0
periodSeconds 0
periodic False
visible True
waitForInput True

All Interfaces | Exported Interfaces



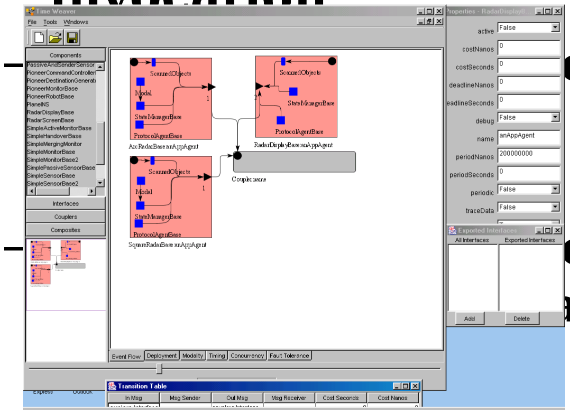
Event Flow View

- **Primary modeling view to represent communications and interactions between components**
- **Multiple communication paradigms can be chosen**
  - **Publish/subscribe event flows**
    - **Unicast->multicast protocols**
  - **Direct function invocation**
  - **calls**
  - **or**
    ant



Components

PassiveAndSenderSensor
PioneerCommandController
PioneerDestinationGenerato
PioneerMonitorBase
PioneerRobotBase
PlaneINS
RadarDisplayBase
RadarScreenBase
SimpleActiveMonitorBase
SimpleHandoverBase
SimpleMergingMonitor
SimpleMonitorBase
SimpleMonitorBase2
SimplePassiveSensorBase
SimpleSensorBase
SimpleSensorBase2



Time Weaver

Properties - RadarDisplay

active	False
costNanos	0
costSeconds	0
deadlineNanos	0
deadlineSeconds	0
debug	False
name	anAppAgent
periodNanos	200000000
periodSeconds	0
periodic	False
traceData	False

Transition Table

In Msg	Msg Sender	Out Msg	Msg Receiver	Cost Seconds	Cost Nanos

Time Weaver Enhancements

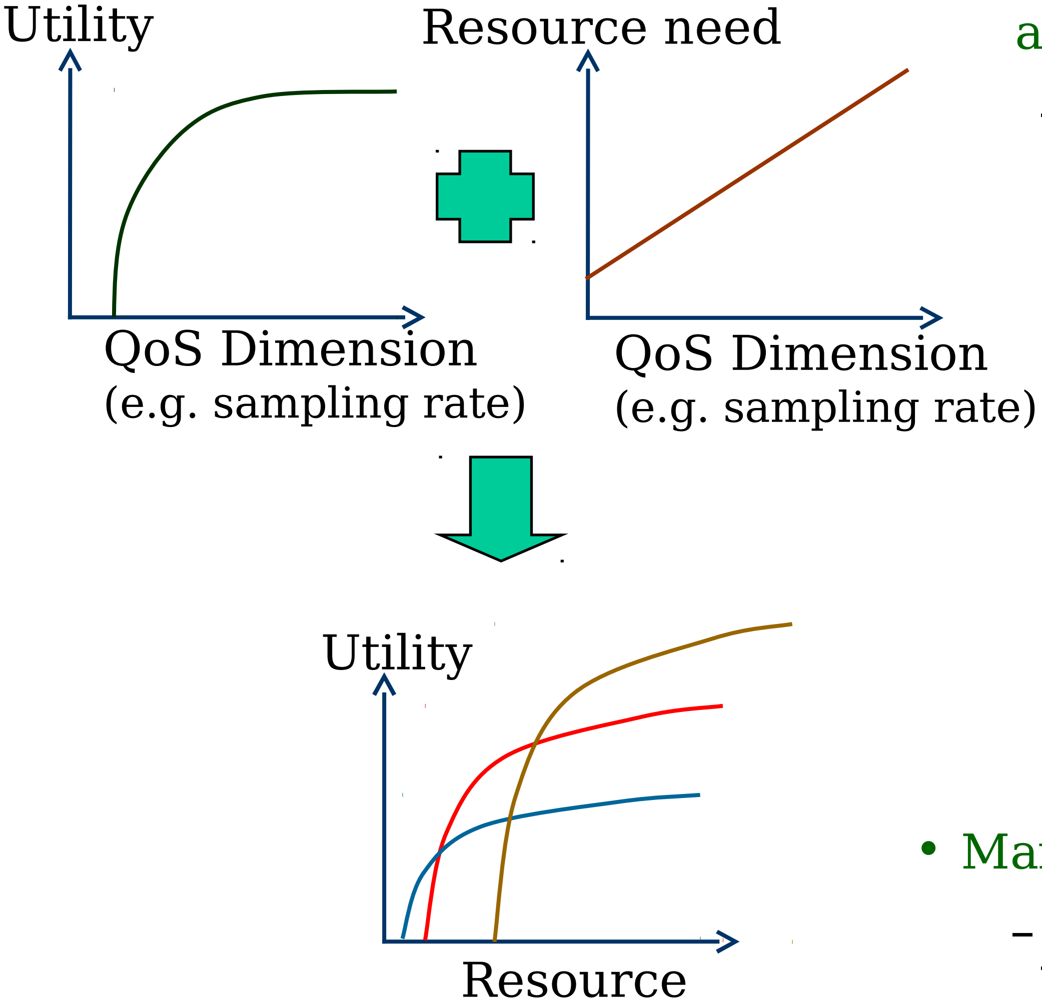
- Compositionality
  - Multiple orthogonal views
    - single view → 4 views
- Multiprocessor modeling
  - End-to-end timing analysis
- Event Modeling and System Modalities
  - Systems often operate in different modes
  - Component modes and system-wide modes
- Modeling
  - Scalability:
    - # of components
      - Fixed: workarounds to serious Java graphics inefficiency
  - File saving:
    - Any and all models can be saved even if not connected
  - Usability speed:
    - Graphics inefficiency fix also fixes usability speeds
- Meta-modeling
  - Need to present a visual interface

Time Weaver Characteristics (2)

***Current* constraint**	• **Cannot *automatically* reverse-engineer systems**
Tool inputs	• **Rational Rose UML Diagrams (class diagram and sequence diagrams)**
Tool outputs	• **TimeWiz hardware & software configuration diagrams** • **Generated RT-Java code that can run on RTOS targets**
Meta-model	• **Port-based objects for inter-component communications** – **Components can be composed from components** • **All inter-component communications happen through “Couplers”** • **Components can have application agents (for functional behavior), state agent and protocol agent.**
Tools interfaced	• **Rational Rose, TimeWiz, Teknowledge UML Interface**

QoS Tradeoff & Optimization

QoS-based Resource Allocation Model (Q-RAM)



- The system consists of multiple applications
  - Each application is characterized by
    - Multiple QoS dimensions
      - Latency, encryption level, availability, …
      - Data frame-rate, data resolution, window size, audio sampling rate, …
    - Multiple resource requirements
      - CPU cycles, disk bandwidth, network bandwidth
  - Utility curves express satisfaction along *each* QoS dimension
- Maximize global system utility by
  - Appropriately allocating finite system resources to applications

Q-RAM Results

- Replication requirements (e.g. for radar processing) create a multiple resource allocation problem
- The older Q-RAM algorithms for multiple resource allocation do not do well with fault-tolerance requirements
  - We have developed two new algorithms: amrmd_dp and amrmd_cm
    - amrmd_dp has higher run-time complexity
- amrmd_cm performs well
  - Admits the most number of tasks (by a significant margin)
    - Performs much better even if the number of processor choices is rather limited
  - Delivers the best utility of known algorithms
  - Has only slightly worse run-time behavior than amrmd1

Future Work

Time Weaver → OSEK

- Time Weaver generates Real-Time Java code and (Real-Time) CORBA interfaces for distributed system execution
  - Tool framework is generic, however
- Goal: Generate C/C++ code for OSEK target
  - OSEK OS, COM and OIL collection studied in detail

- Separate local and non-local interaction interfaces
- Multiplicity of

- Minimum portability requirements & conformance classes
  - 8 basic/16 extended priorities: footprint vs. portability
  - 1 alarm: single periodic application task
- Tasks: Fixed priority preemptive scheduling
  - Multiple local dispatch/interaction mechanisms
  - Explicit activation, events: Task topology in application code
  - Non-uniform dispatch request queuing & task initialization
- Events: Binary task flag
  - Lossy commmunication & dispatch mechanism
  - Non-deterministic non-FIFO priority level task queuing
- Resources
  - Stateless task non-preemption via priority ceiling protocol
  - Unnecessary basic, internal, linked resource mechanisms
- Alarms

Time Weaver→OSEK OIL 2.3 System Modeling

- **Task and communication architecture**
  - **Single processor only**
  - **Multiple task configurations: Incomplete application mode support**
  - **Portability limitation: Task topology embedded in application code**
  - **Not quite MetaH - a DARPA EDCS/DASADA technology & emerging SAE AADL standard**
- **Scheduling analysis**
  - **Periodicity via alarm initialization: changeable by application code**
  - **Incomplete task dispatch information: explicit activation excluded**
  - **Missing timing properties**
- **Auto-generation of code**
  - **Runtime executive with application code as plug-in**

COM2.2.2 and 3.0 Common features

- **Local and network communication**
- **Queued and unqueued messages**
- **Direct and periodic transfer**
- **Topology in OIL [illegible] in**
- **[illegible] local communication semantics**

Proliferation of mechanisms in support of runtime efficiency

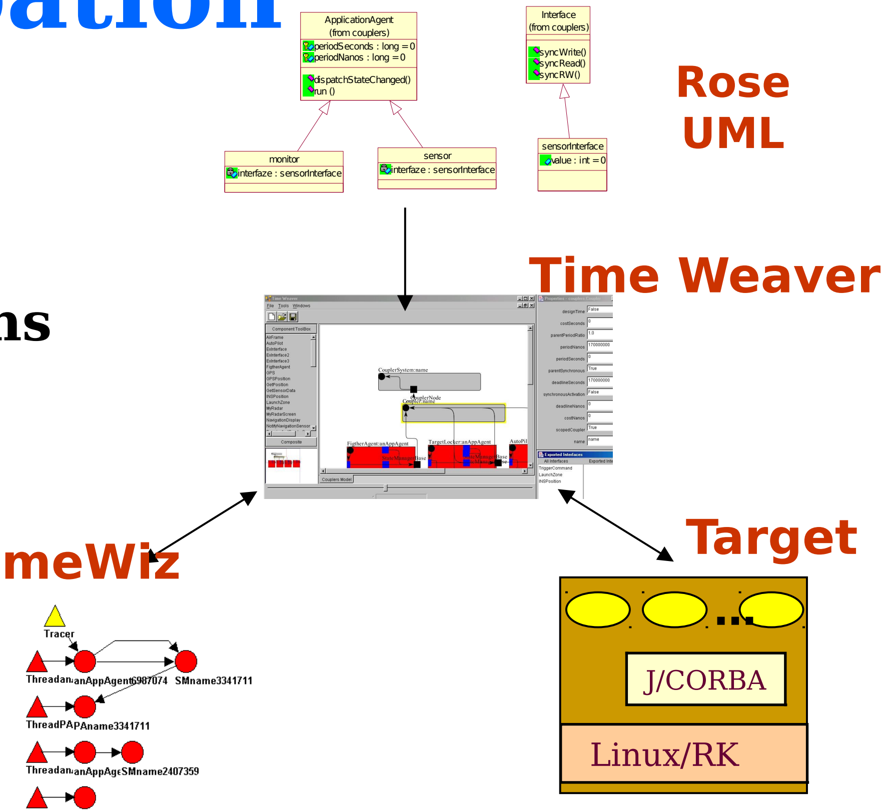
COM 2.2.2	COM3.0
1:n communication	m:n communication
Send, xfer, receive	Pack, send, xfer, receive
Single queue receiver	Multiple queue receivers
Send/Rec without copy	Zero length msg: event alternative
Message initialization gaps	Msg filter: zero length msgs excluded

Avionics OEP Participation

Roles:

- End-to-end timing & schedulability analysis
- Concurrency modeling
- Inter-Process Communications modeling
- Java and CORBA code generation
- Tool interoperability

- **Mid-Term Experiments**
  - Timing and schedulability analysis
  - Concurrency & comm optimization
  - Configurator
  - Interfacing with UML diagra
  - Tool interoperability



TimeWiz

Also worked with STRIVE project with Lockheed Martin

- **Expected technology transition t o F-35**
- **Time Wiz → F-35**

Automotive OEP Participation

- **End-to-end Timing and Schedulability Analyses**
  - **Processor allocation**
  - **QoS modeling based on application utility functions (reward as a function of QoS)**
- **Automotive OEP**
  - **OSEK analysis and optimization**
  - **Technical Points of Contact:**
    - **Anouck Giraud at Berkeley**
    - **Bill Milam at Ford**

Interactions with Teknowledge

- Annotate Rose objects:
  - Represent object models in Rose adding tagged values to represent additional data.
- Rose←→TimeWeaver
  - Feed Time Weaver's analysis results back to Rose
- Incremental analysis
  - Tailor Time Weaver in such a way that it can do partial model checking based on changes in Rose only.
  - Longer term

Project Status Update (1)

- Technology Update
  - Analyzable and reusable software component framework
  - Multiple semantically orthogonal views
  - QoS requirements can be specified and optimized with given resources
  - Concurrency modeling and inter-process communicaiton modeling
  - Event dependency modeling and analysis capabilities
  - Consistent XML interface to all tools of interest
  - Can work with CORBA code
- Tool Update
  - Inter-operability between tools
    - Boeing OEP requirements: ACL, Configurator and model representation
  - Time Weaver Release: http://www.cs.cmu.edu/~rtml
    - Completely new (better and friendlier) GUI for tool

Status Update (2)

- OEP Participation
  - Successful mappings to avionics context
    - Detailed feedback from Boeing, UCB and Lockheed Martin experiments
  - OSEK studies in automotive context
    - Currently choosing from multiple options supported by OSEK

6-month Project Plans

- Avionics OEP
  - Apply feedback from Boeing and Lockheed Martin evaluations
    - Scalability and usability verification
    - Extensive documentation
  - Scalability support: aggregation and encapsulation
  - Fault-tolerance modeling, modality analysis and C/C++ support
- Automotive OEP
  - Model ETC dual-processor and cruise control environments
    - Analyze and optimize OSEK-based implementations
- Specific performance goals
  - Efficient support for automotive OEP and platforms
  - Instrumentation support
  - Lower processor and network utilization by automated optimization of # of threads, and communications.

Milestones and Schedule

1. Concurrency and Timing Analysis
   - 2QFY01: Timing and schedulability analysis
   - 3QFY02: End-to-end timing analysis
2. Constraint-Based Composition
   - 2QFY02: Event dependency modeling
   - 1QFY03: Replication modeling and replication support
3. Code Generation
   - 2QFY02: Real-Time Java code generation
   - 3QFY02: C++/Real-Time CORBA
   - 4QFY02: OSEK code generation
4. Model-Checking
   - 1QFY02: Transmission control property verification
   - 1QFY02: Worst-case execution time determination
5. QoS Tradeoff Support
   - 1Q02: QoS Specification and tradeoffs
   - 3Q02: Scalability
6. Tool Integration and Collaboration with OEPs
   - 2QFY02: Boeing and LM OEP mid-term experiments
   - 4QFY02-3QFY03: Avionics and Automotive OEP

Technology Transition/Transfer

- DoD contractors
  - Boeing
  - Lockheed Martin
  - Raytheon
- OMG Standardization
  - Real-time software component models
  - Interoperability with Real-Time UML
- TimeSys Corporation (www.timesys.com)
  - Commercial vendor of the TimeWiz timing analysis tool and TimeSys Linux with hard real-time and QoS support.
    - TimeWiz extensions from CMU to be returned to TimeSys for commercialization